(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 069 532 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
17.01.2001 Bulletin 2001/03

(51) Int. Cl.[7]: **G06T 15/00**

(21) Application number: **00113247.1**

(22) Date of filing: **21.06.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **15.07.1999 US 353882**

(71) Applicant:
MITSUBISHI DENKI KABUSHIKI KAISHA
Tokyo 100-8310 (JP)

(72) Inventors:
- **Lauer, Hugh C.**
  Concord, MA 01742 (US)
- **Seiler, Larry D.**
  Boylston, MA 01505 (US)
- **Knittel, James M.**
  Groton, MA 01450 (US)
- **Correll, Kenneth W.**
  Lancaster, MA 01523 (US)
- **Gasparakis, Charidimos E.**
  Acton, MA 01720 (US)
- **Shimha, Vikram**
  Lexington, MA 02173 (US)
- **Bhatia, Vishal C.**
  Arlington, MA 02474 (US)

(74) Representative:
Pfenning, Meinig & Partner GbR
Mozartstrasse 17
80336 München (DE)

(54) Multi-pass volume rendering pipeline

(57) A multi-stage pipeline renders a volume data set stored in a volume memory as samples. The pipeline includes a first stage reading the samples from the volume memory. A second stage renders the read samples, and a third stage writes the processed samples to the volume memory. The first, second and third stages use first rendering parameters during a first rendering pass and second rendering parameters during a second rendering pass to enable multi-pass volume rendering.

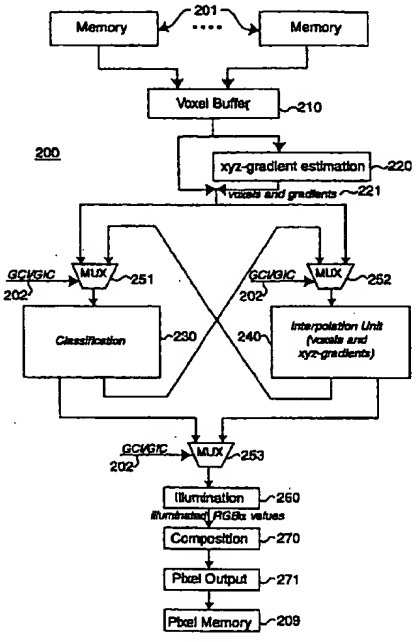

FIG. 2

EP 1 069 532 A2

Description

FIELD OF THE INVENTION

[0001] This invention relates generally to volume rendering, and more particularly, to a hardware pipeline wherein a volume data set is processed multiple times.

BACKGROUND OF THE INVENTION

Introduction to Volume Rendering

[0002] Volume rendering is often used in computer graphics applications where three-dimensional data need to be visualized. The volume data can be scans of physical or medical objects, or atmospheric, geophysical, or other scientific models where visualization of the data facilitates an understanding of the underlying real-world structures represented by the data.

[0003] With volume rendering, the internal structure, as well as the external surface features of physical objects and models are visualized. Voxels are usually the fundamental data items used in volume rendering. A voxel is a data item that represents a particular three-dimensional portion of the object or model. The coordinates (x, y, z) of each voxel map the voxels to positions within the represented object or model.

[0004] A voxel represents some particular intensity value of the object or model. For a given prior art volume, intensity values can be a specific one of a number of different parameters, such as, density, tissue type, elasticity, or velocity. During rendering, the voxel values are converted to color and opacity (RGB.) values, according to the intensity values, which can be projected onto a two-dimensional image plane for viewing.

[0005] One frequently used technique during rendering is ray-casting. A set of imaginary rays are cast through the array of voxels. The rays originate from a viewer's eye or from an image plane. The voxel values are re-sampled to points along the rays, and various techniques are known to convert the sampled values to pixel values. Alternatively, voxel values may be converted directly to RGB. voxels, which are then re-sampled along rays and accumulated to pixel values. In either case, processing of the volume data may proceed back-to-front, or front-to-back.

Rendering Pipeline

[0006] Volume rendering can be done by software or hardware. In one hardware implementation, the hardware is arranged as a multi-stage pipeline, see U.S. Patent Application 09/190,643 "Fast Storage and Retrieval of Intermediate Values in a Real-Time Volume Rendering System," filed by Kappler et al. on Nov. 12, 1998.

[0007] Figure 1 illustrates a pipeline 100 wherein voxel values are stored in a voxel memory 101. The voxel values are first read into a voxel buffer 110 of the pipeline as slices. The z-components of the gradients are estimated in stage 115 by taking central differences between voxels of different slices. Then, both the voxel values and the z-gradients are passed to an interpolation stage 120 that calculates these values at sample points along rays. Next, the x- and y-components of the gradients are calculated from the interpolated sample values in stage 130. These, along with the sample values and the interpolated z-gradients are then passed to a classification stage 140, and then a shading stage 145, where an illumination process is applied to produce the RGB. values representing the illuminated samples. Finally, the illuminated samples are combined along rays in an compositing stage 150 to produce pixel values for the base plane stored in a pixel memory 109.

[0008] That pipeline structure suffers because the it is only possible to render voxels in a single pass. It is desired to process the voxels in multiple rendering passes.

SUMMARY OF THE INVENTION

[0009] The invention provides a volume rendering pipeline including a plurality of processing stages. The stages can include a gradient estimation stage, an interpolation stage, a classification stage, an illumination stage, and a compositing stage. The stages are connected to each other by multiplexers.

[0010] The pipeline is coupled to a volume memory storing a volume data set as samples. Initially, the samples can be voxels. A first stage of the pipeline reads the samples from the volume memory. A second stage renders the samples. A third stage of the pipeline writes samples to the volume memory. The first, second and third stages use first rendering parameters during a first rendering pass and second rendering parameters during a second rendering pass to enable multi-pass volume rendering.

[0011] The volume rendering pipeline is also coupled to a pixel memory for storing pixels generated by the compositing stage of the pipeline during a last rendering pass.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012]

Figure 1 is a block diagram of a prior art rendering pipeline;

Figure 2 is a block diagram of a configurable rendering pipeline;

Figure 3 is a pipeline with interpolation before classification;

Figure 4 is a pipeline with classification before interpolation;

Figure 5 is a block diagram of a flexible format voxel.

Figure 6 is a block diagram of a field format register;

Figure 7 is a block diagram of a voxel format register;

Figure 8 is an example formatted voxel;

Figure 9 is a block diagram of a configurable multi-pass rendering pipeline connected to a volume memory.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0013] Figure 2 shows a top-level block diagram of a configurable rendering pipeline 200 according to the invention. The pipeline 200 takes samples or voxels as input from a voxel memory 201, and stores pixels as output in a pixel memory 209. The voxels or samples are read into a voxel buffer 210 a volume slice at the time. Gradients are estimated for *xyz*-components in stage 220. Here, in contrast with the prior art, all gradient components are estimated on voxel or sample values, not on interpolated samples.

[0014] The voxels 221 are classified in stage 230. Interpolation of voxels and gradients takes place in stage 240. Gradient estimation and interpolation are linear operations and therefore can be interchanged. As an advantage of the invention, the application can determine the order in which the voxels and gradients are processed by the various stages. The reason for two different processing orders is a follows.

[0015] Scanners acquire sampled data by making physical measurements which typically represent an integral over a small region of space. Adjacent voxels often represent the integrals of overlapping regions. In these cases, it is more accurate to interpolate voxels first, then classify the result. In particular, a voxel at a boundary of, say, two different tissues is likely to represent an average of the physical properties of each tissue. When interpolating, all that is done is moving the points of average. To make this useful, the classification function should present a continuous transition from between the colors and opacities assigned to the two tissue types.

[0016] Sometimes, sampled data may be pre-segmented (manually or automatically) into distinct materials, e.g., bone, muscle, cartilage, etc. Given such data, it would be inappropriate to interpolate between two differently identified tissues to obtain a third tissue that is not really there. In these cases, it is better to classify the voxels first, then interpolate the resulting colors. This is not so much an average of the physical properties of adjacent tissue types, but rather a blend of the colors at their boundaries.

[0017] Therefore, the pipeline 200 includes a multiplexer 251 connecting the output from stage 240 to the input of stage 230. Similarly, a multiplexer 252 connects the output of stage 230 to the input of stage 240. Multiplexer 253 selects the output from either stage 230 or stage 240.

[0018] The order of connection of the stages depends on a selection signal (GCI/GIC) 202 to the three multiplexers 251-253. In either case, interpolated gradients and interpolated samples are passed to the illumination stage 260. Illuminated RGB. values are composited in stage 270, and output pixels 271 are stored in the pixel memory 209.

[0019] In the pipeline 200, the output of the gradient estimate stage 220 is the three components of gradients. These are passed to both the classification stage and the interpolation stage through the set of multiplexers. The output of each of these stages is each passed to the input of the other stage and also to the illumination stage 260 through another multiplexer.

[0020] By appropriately selecting the three multiplexers 251-253, it is possible to do classification before interpolation, or interpolation before classification, or only interpolation, or only classification. These different modes of operation is now described in greater detail.

GIC — Gradient, Interpolation, Classification

[0021] Figure 3 illustrates the rendering process for the GIC mode. This mode can be selected by signal 202. In the GIC mode, gradients are first estimated, then interpolation takes place on gradients and voxels in parallel, followed by classification of the voxels.

[0022] Rendering proceeds a section at a time, that is, in groups of 32×32 rays. In order to process a section of rays, slices of voxels are read into the voxel buffer 210 two slices at the time. Each slice has only the voxels needed for that section. The number of voxels per slice depends upon the view direction and resampling frequency. The maximum number of voxels needed is 37×37 voxels.

[0023] Reading voxels into the voxel buffer 210 can consumes the full bandwidth of the destination slices. Therefore, at any given time, one pair of slices is dedicated to receiving new voxels, while another two pairs of slices are dedicated to supplying the next stages of the pipeline, i.e., the gradient estimation 220 and the voxel interpolation stages 240.

[0024] In Figure 3, slices labeled *s +1* and *s +2* are receiving new voxels, while slices *s*, *s-1*, and *s-2* are providing voxels to the subsequent stages. Slice *s-3* was used during a previous slice and is currently empty. When the processing of slice *s-2* is complete, slice *s-2 is* also marked as empty. Then, slices *s-2* and *s-3* will become the new destination for voxel reads, while the processing modules will start taking their input from slices *s +1*, *s*, and *s-1*.

[0025] The gradient estimation stage 220 estimates

gradients at voxel points in slice *s-1* from voxel values in slices s and *s-2*. These gradients are then stored in a two slice gradient buffer 225 for interpolation by unit 228. Gradients for voxel points in slice *s-2* had been stored in the gradient buffer during a previous slice iteration.

[0026] Next, voxel interpolation 240 and gradient interpolation 228 proceed concurrently. That is, a slice of samples is obtained by interpolating voxel values from voxel slices *s -1* and *s -2*. Gradients at a sample points in the current slice of samples are obtained by interpolating the gradients of gradient slice *s -1* and gradient slice *s -2* in the gradient buffer. More than one slice of samples and interpolated gradients can be obtained from the same pair of voxel and gradient slices.

[0027] The interpolated voxel values are next applied to the classification stage 230, which converts the values to interpolated RGBα color values. The RGBα values and the gradients are next supplied to the illumination stage 260, and finally to the compositing stage 270.

[0028] In an optional mode, gradients are extracted directly from, for example, three voxel fields, instead of using the gradient estimation stage 220 to estimating gradients from any voxel field.

GCI - Gradient, Classification, Interpolation

[0029] Figure 4 illustrates the rendering algorithm for the GCI mode, also selected by signal 202. The voxel buffer 210 is filled as described above, and the gradient estimation process is also the same. The difference is in the voxel classification and interpolation side. In particular, raw voxels are taken from the voxel slice buffer, that is slice *s -1*, and immediately converted to RGBα values by the classification stage 230. These are then stored in a two slice RGB. buffer 235. Note, the RGB. buffer 235 is parallel to the two slice gradient buffer 225.

[0030] The two slices are then input to the RGB. interpolation stage 240, where interpolated RGB. values are produced in parallel with interpolating gradients. These are then applied with interpolated gradients to the illumination and compositing stages to produce the pixel output 271 of the base plane image. These last two modules are identical to those of Figure 2-3.

[0031] One additional optional mode is also possible. In this optional mode, multiple, for example, four, fields of the voxel are interpreted as raw RGB. values. In this case, the gradient may be estimated from the alpha (.) field.

[0032] It should be noted that for some volume data sets and rendering modes it may be best to estimate gradient in a later stage of the pipeline, for example, after classification.

[0033] It should be apparent that the multiplexers 251-253 can be replicated for other stages. Stages can selected and connected in a number of different orders to provide a reconfigurable rendering pipeline. Some stages can be de-selected to not process the volume data at all, for example, for some renderings gradient estimation and illumination may be skipped.

Flexible Voxel Format

[0034] As shown in Figure 5, a voxel 500 as used by the configurable rendering pipeline 200 includes a plurality of fields (V_1, ..., V_n) 501-209. Each of the fields 501-209 can be specified as an offset and width in the voxel 500. Any of the fields can overlap as shown for fields V_1 and V_2. The fields can be enumerated in any order.

[0035] The fields describe different attributes of a represented three-dimensional object or model. For example, if the object is a human head, then the fields 501-209 can respectively store intensity values acquired from CT, MRI, PET, SPECT, and ultrasound scans. i.e., each voxel may store five different scan intensity values in a single volume representation. Alternatively, the scans can be stored as multiple separate volumes wherein each voxel contributes one field.

[0036] Some fields can be category fields related to the way the volume is segmented, e.g., manual, semi-automatic, or automatic segmentation. In medical applications, segmentation can categorize bone, tissue, etc., In physical applications, segmentation can identify sub-assemblies or other parts to be rendered in a particular way. For physical models, the fields can store state variables used in scientific visualization, e.g., pressure, velocity, angular momentum, elasticity, density, temperature, and viscosity. For any volume data set, the fields can also store RGB. values, depth, 3D stencil, shadows, fog, voxelized and embedded geometry volumes, or gradients.

[0037] For a multi-field voxel 500 according to our invention, the user can specify which fields to use for gradient calculations. For each component of the gradient, we can specify which of the voxel fields to use for that component.

[0038] For multi-field visualization, it is usually desirable to interpolate fields within voxels separately. Furthermore, each field within the voxel can have a different interpolation function applied, e.g., tri-linear for intensity fields, and nearest neighbor interpolation for category fields. The flexible voxels as described herein enable a common framework for treating all special cases of voxel formats in a uniform fashion.

Field Format Register

[0039] Figure 6 shows a field format register 600 that enables multiple field voxels according to the invention. In one embodiment, fields of voxels are defined by descriptors of the field format register. This is an 8-bit (7:0) descriptor defining the size of the field (in 4-bit nib-

bles) 610, the position of the field within its voxel 620, pixel (also in 4-bit nibbles), and what to do (control 630) when the field is a different size than specified for its intended use.

[0040] The control bits define how a field may adapted to fit the data path of the pipeline through which the field will pass. The field can be changed by repeating fraction arithmetic, or by adding or removing bits from either the most significant or least significant end.

Control = 0: the field of the raw voxel is treated, as an unsigned repeating fraction representing a number in the range [0...1]. To expand or shrink the repeating fraction to fit the data path, repeating fractional arithmetic is applied to scale and round, thereby representing the number with fewer or more bits of precession. Repeating fraction number representation is described in greater detail below.

Control = 1: the field of the voxel is treated as a signed repeating fraction in 1 the range [-1...+1].

Control = 2: the field of the raw voxel is expanded or truncated in its least significant bits to fit the data path. The most significant bits are preserved.

Control = 3: the field of the raw voxel is expanded or truncated in its most significant bits to fit the data path. The least significant bits are preserved.

Repeating Fraction Number Representation

[0041] Many graphics applications use a fixed width binary number to represent color, transparency, or other parameters that have values in the range zero to one, inclusive.

[0042] Let R be the number of bits in the binary number and let V be the unsigned binary value stored in these bits. Then $F = V / (2^R - 1)$ is a rational number in the range $[0..1]$. That is, when V equals zero, F equals zero, and when V equals its largest possible value, $(2^R - 1)$, F equals one. This representation is well known in the prior art. For example, the OpenGL Specification refers to it as a special kind of fixed point representation.

[0043] To clearly distinguish the representation described herein from ordinary fixed point representation, the term "repeating fractions" is used. The name term derives from the fact that expanding F into a fixed point binary fraction produces $0.VVVVVV...$, that is, a binary fraction that repeats the R-bit value V infinitely to the right of the binary point.

[0044] Repeating fractions can be represented with more than R bits and can even be signed. In that case, R is the "repeating precision" of the number, since R defines the implicit scale factor $(2^R - 1)$. This allows F to have values outside the range $[0...1]$. In this case, the binary fixed point representation consists of an integer value followed by an R-bit infinitely repeating binary value. Repeating fractions with the same precision may be added and subtracted in the same way as ordinary integers.

[0045] Other arithmetic operations, including changing the repeating precision, may be performed by first computing F for each repeating fraction, performing normal arithmetic, and then multiplying the resulting value by $(2^R - 1)$ for the repeating precision R of the result. More efficient forms exist for operations on repeating fractions. For example, doubling the repeating precision from R to $2R$ simply requires computing $V + (V<<R)$.

Voxel Formats

[0046] The pipeline according to the invention allows a wide range of input formats of voxels. Input voxels can be 8, 16, 32, or larger bit quantities. Fields can be 4, 8, 12, or 16 bits in width, and are aligned on a 4-bit boundary within the voxel. All voxels fields are scaled to fit their data paths, which are typically twelve bits wide.

[0047] The format of a voxel is described in a voxel format register 700. An example of the format of a 32-bit voxel 800 is illustrated in Figure 8. In Figure 7, the format register for the example voxel, *Field0* occupies bits 11:0, *Field1* occupies bits 23:20, *Field2* occupies bits 31:24, and *Field3* overlaps *Field0* and occupies bits 15:8. For the example shown in Figure 8, the descriptions of the fields in field format register are as follows:

Field3: Size = 1	Position = 2,
Field2: Size = 1	Position = 6,
Field1: Size = 0	Position = 5, and
Field0: Size = 2	Position = 0.

Advantages

[0048] Having flexible format voxels, and a reconfigurable pipeline enable a number of advantages. For example, one of the fields in a voxel can be used for category bits. These are extra bits in voxels that identify the voxel as part of some particular tissue, sub-assembly, or other partition of the volume. They are not interpolated, but they do contribute to the assignment of RGB. values to voxels. When category bits are used, classification usually precedes interpolation.

[0049] The flexible voxel format allows gradients to be estimated from any selected field of the voxel. In these cases, a convolution kernel, such as a central difference, is applied to the selected field of each voxel to obtain the x, y, and z-components of the gradient of that voxel. These gradients are then interpolated, in parallel with raw or classified voxels, in order to obtain gradients at sample points. These are then applied, along with RGB. values to the illumination stage 260, and finally to the compositing stage 270.

[0050] In some applications and for some volumes,

it may be better to use gradients that are determined in some other manner, instead of estimating them from the fields of voxels on the fly. Flexible format voxels accommodate this mode. In particular, gradients may be precomputed for the voxels with a high level of precision. These precomputed gradients can be stored in one of the fields of the voxels. These precomputed gradients can bypass the estimation stage, but are interpolated and applied in the illumination stage 260 just as the gradients estimated on the fly.

[0051] In a further variation, precomputed gradient are applied to a convolution kernel for gradient estimation. This has the effect of taking the second partial derivative of the volume data set. By the same technique, higher order partial derivatives may be obtained. Such derivatives are useful for extracting weak surfaces in the volume data set. As described for Figure 9 below, this can be done by passing the volume data set through the pipeline (1-3) multiple times in a "multi-pass" operation. Each pass processing the volume data set with different set of rendering parameters.

[0052] The pipeline 200 processing flexible format voxels also admits volume data sets that are preclassified and presented as "RGB. " voxels. Two alternatives are useful. In one, gradient estimation and the classification and shading are be skipped entirely, and the voxels simply interpolated. In the other alternative, gradients are estimated from alpha values or a luminance function, and shading the RGBa values is done according to the illumination function. Note, the pipeline 200 according to the invention is also capable of interpolating both voxel-gradient values and RGB. values.

Multi-pass Volume Rendering

[0053] The rendering pipeline 200 can also write out a volume data set that has passed through some, but not necessarily all, of the stages of the pipeline 200 back to the volume memory, see Figure 9. In an example use, the configured pipeline renders a set of voxels into classified (color and opacity weighted RGBα) samples. Instead of composited along rays, the colored RGBα samples are stored back to the memory 201 as RGBα samples in a three-dimensional volume data set. Then, the volume is rerendered with a different set of rendering parameters. This process can repeat until a final volume is present in memory 201. The final accumulated values are then rendered one final time to generate an image in the pixel memory 209.

[0054] This technique enables a number of features, such as fast resampling and second order gradient estimation. With fast resampling, the volume data set can be resampled to a different resolution using the speed and power of the volume rendering pipeline and volume memory, instead of relying upon the host processor and software. Higher order gradient estimation can be used to extract weak surfaces.

[0055] Multi-pass rendering can be used to produce complex shadows on the volume. One pass is needed for each light source. Each pass is accumulated in the output volume data set, and then a final pass interpolates the results and projects them onto the base plane. In multi-pass rendering, the volume data set output from a current pass is combined with an already existing volume data set in the memory. This sometimes requires a read-modify-write operation rather than a simply write operation.

Multi-Channel Rendering

[0056] The pipeline 200 can also process multi-channel data sets. Some scanning techniques, such as, ultrasound and seismic applications, have data of more than one type, each with its own classification. During rendering these data sets are be superimposed on each other, and combined, voxel-by-voxel.

[0057] More particularly, the compositing stage 270 is able to operate in either of two modes. In a first mode, the RGBα values are combined with previously stored pixel values, and in a second mode, a ray of RGBα values is accumulated, and the result is combined with a previously stored pixel value from some previous pass.

Pipeline General Structure

[0058] Generally, the relation of the pipeline 200 and the memory 201 can be as shown in Figure 9. Here, the stages ($stage_0$, $stage_1$, ..., $stage_n$) 1-3 of the pipeline 200 are connected to each other by multiplexers 4 so that for a particular rendering application, the stages are ordered by a select signal 5.

[0059] The input to the pipeline 200 is typically, at least during a first pass of a multi-pass rendering, a raw volume data set stored in a volume memory 6. The voxels are read by the first stage 1. The voxels read are processed by the second stage 2. This stage can include interpolation, gradient estimation, classification, illumination, and compositing stages as described above. Any of the stages can be selected or deselected, and the order of processing through the selected stages can vary. The output of the pipeline is a modified volume data set according to rendering parameters. The output can be written by the third stage 3. Individual data items of the volume data set passed between the pipeline and the memory are flexible format samples or voxels 7. The volume data set can be processed by multiple passes, each pass using different rendering parameters for the reading, processing, and writing.

[0060] It is to be understood that various other adaptations and modifications may be made within the spirit and scope of the invention. Therefore, it is the object of the appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.

Claims

1. An apparatus for rendering a volume data set stored in a volume memory as samples, the apparatus comprising:

   a first stage reading the samples from the volume memory;
   a second stage processing the read samples; and
   a third stage writing the processed samples to the volume memory, the first, second and third stages using first parameters during a first pass and second parameters during a second pass to enable multi-pass volume rendering.

2. The apparatus of claim 1 wherein the first, second and third stages are implemented on a single semiconductor device.

3. The apparatus of claim 1 wherein the samples are voxels during the first pass.

4. The apparatus of claim 1 further comprising:

   a pixel memory storing pixels generated by a compositing stage during a last pass.

5. The apparatus of claim 1 wherein the samples are interpolated at a first resolution during the first pass and at a second resolution during the second pass.

6. The apparatus of claim 1 further comprising:

   an illumination stage shading the samples using a first light source during the first pass and a second light source during the second pass.

7. The apparatus of claim 1 wherein the processed samples are combined with the samples stored in the volume memory during the writing.

8. The apparatus of claim 1 wherein first order gradients of the samples are determined during the first pass and second order gradients of the samples are determined during the second pass.

9. The apparatus of claim 1 wherein the samples are classified as color and opacity weighted samples during the first pass and the color and opacity weighted samples are illuminated during the second pass.

10. A method for rendering a volume data set stored in a volume memory as samples, the apparatus comprising the steps of:

    reading the samples from the volume memory;
    processing the read samples; and
    writing the processed samples to the volume memory, the reading, processing and writing first parameters during a first pass and second parameters during a second pass to enable multi-pass volume rendering.

11. The method of claim 10 wherein the samples are voxels during the first pass.

12. The method of claim 10 further comprising the step of:

    storing pixels generated by a compositing stage in a pixel memory during a last pass.

13. The method of claim 10 wherein the samples are interpolated at a first resolution during the first pass and at a second resolution during the second pass.

14. The method of claim 10 further comprising the step of:

    shading the samples using a first light source during the first pass and a second light source during the second pass.

15. The method of claim 10 wherein the processed samples are combined with the samples stored in the volume memory during the writing.

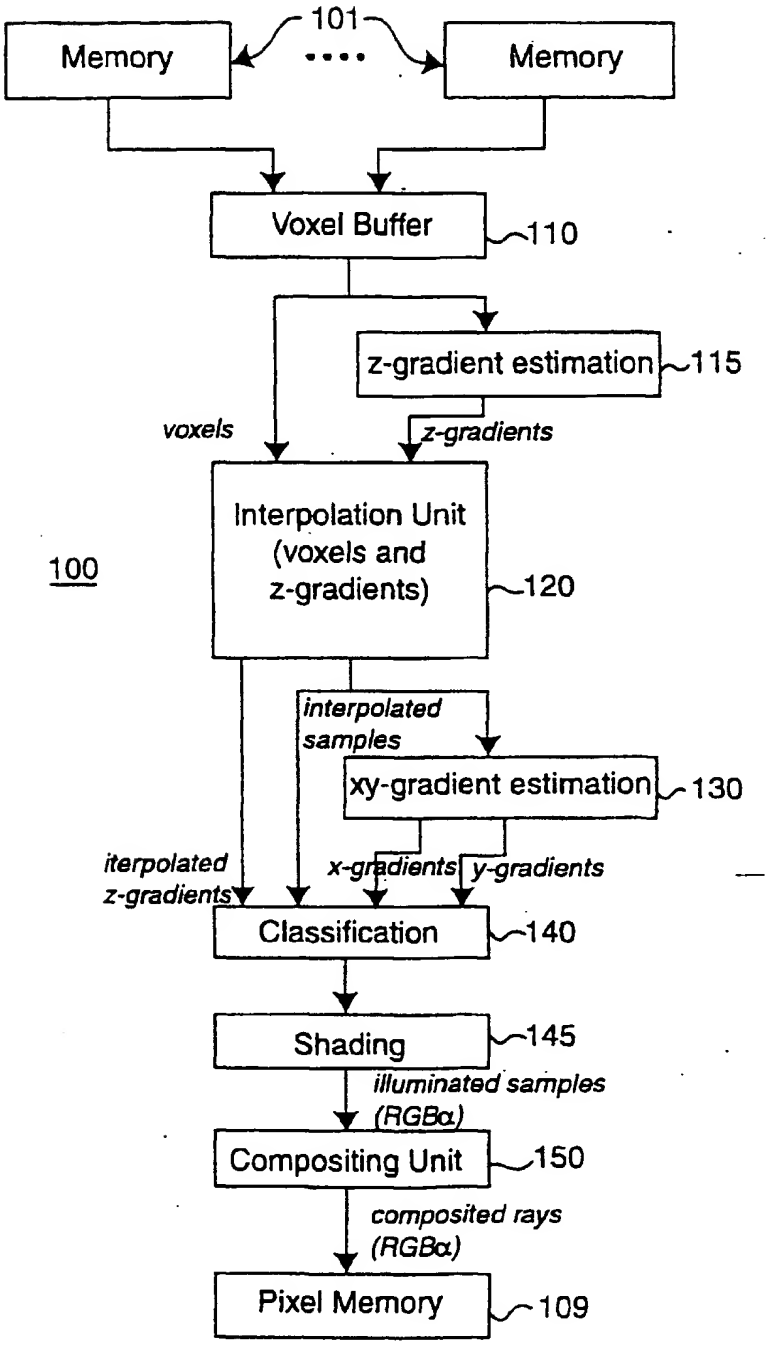

FIG. 1

PRIOR ART

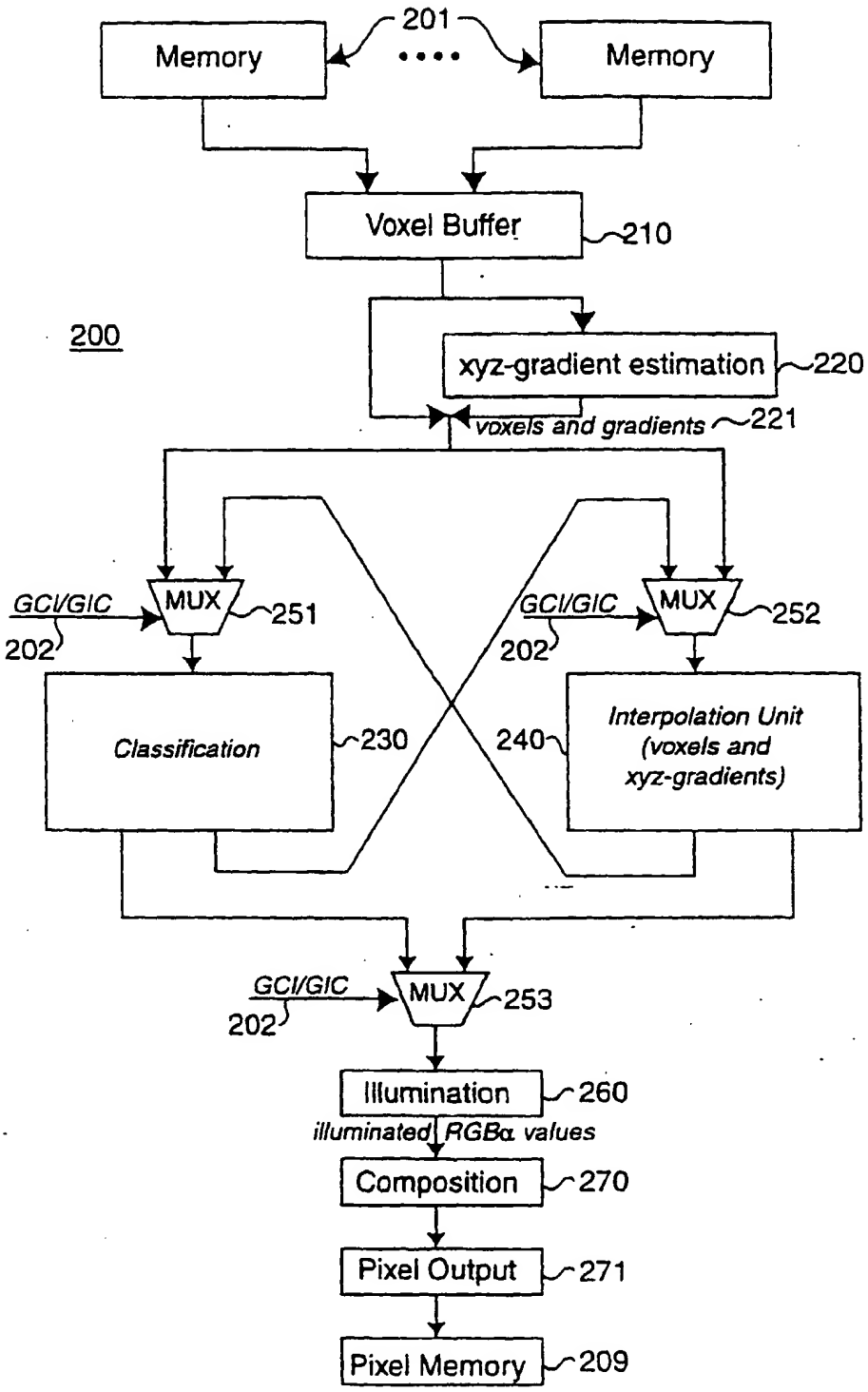
201
Memory
Memory
Voxel Buffer
210
200
xyz-gradient estimation
220
voxels and gradients
221
GCI/GIC
202
MUX
251
GCI/GIC
202
MUX
252
Classification
230
240
Interpolation Unit
(voxels and
xyz-gradients)
GCI/GIC
202
MUX
253
Illumination
260
illuminated RGBα values
Composition
270
Pixel Output
271
Pixel Memory
209

FIG. 2

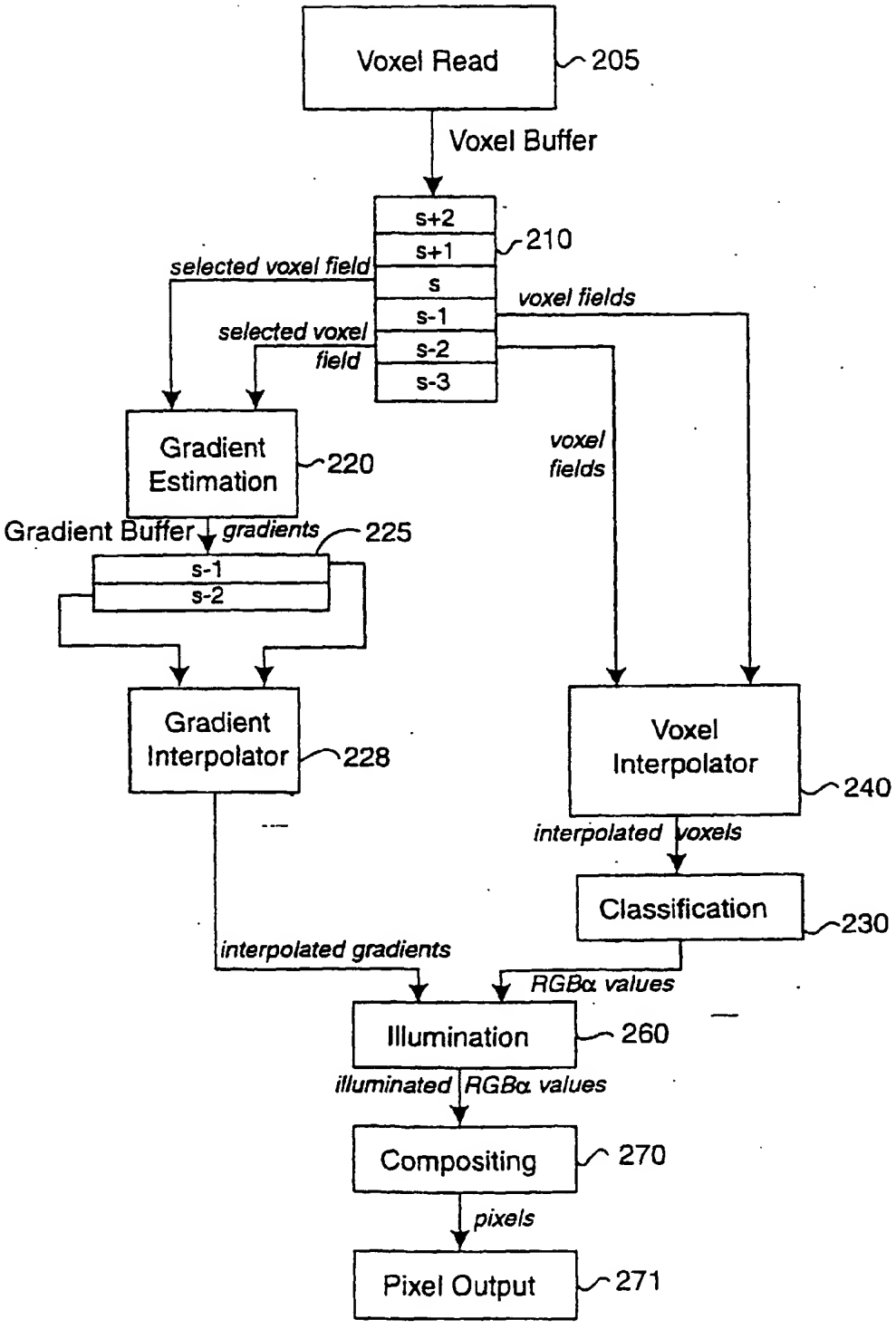
Voxel Read
205
Voxel Buffer
s+2
s+1
s
s-1
s-2
s-3
210
selected voxel field
selected voxel field
voxel fields
Gradient Estimation
220
voxel fields
Gradient Buffer
gradients
225
s-1
s-2
Gradient Interpolator
228
Voxel Interpolator
240
interpolated voxels
Classification
230
interpolated gradients
RGBα values
Illumination
260
illuminated RGBα values
Compositing
270
pixels
Pixel Output
271

FIG. 3

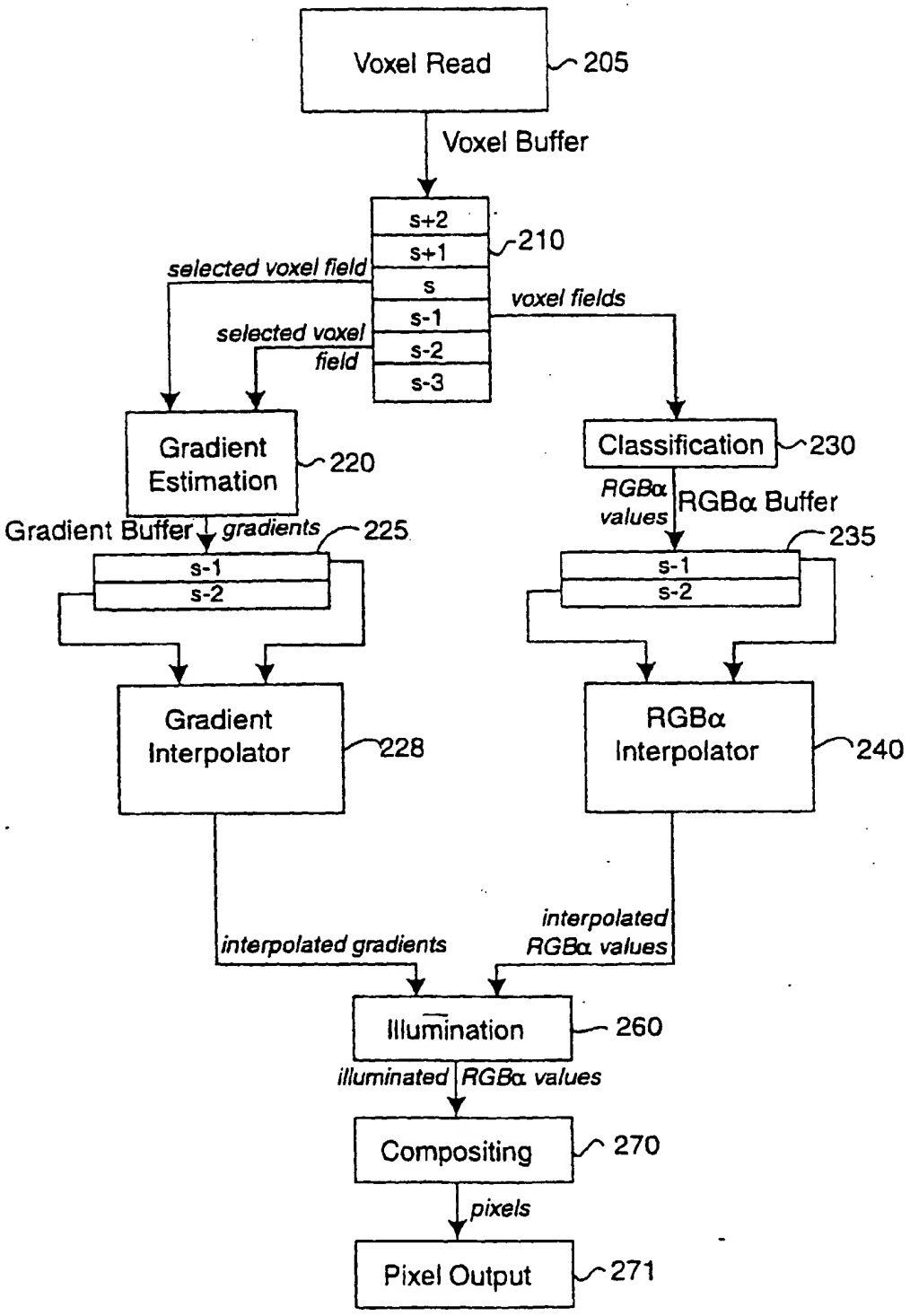
Voxel Read
205
Voxel Buffer
s+2
s+1
s
s-1
s-2
s-3
210
selected voxel field
selected voxel field
voxel fields
Gradient Estimation
220
Classification
230
RGBα values
RGBα Buffer
Gradient Buffer
gradients
225
235
s-1
s-2
s-1
s-2
Gradient Interpolator
228
RGBα Interpolator
240
interpolated gradients
interpolated RGBα values
Illumination
260
illuminated RGBα values
Compositing
270
pixels
Pixel Output
271

FIG. 4

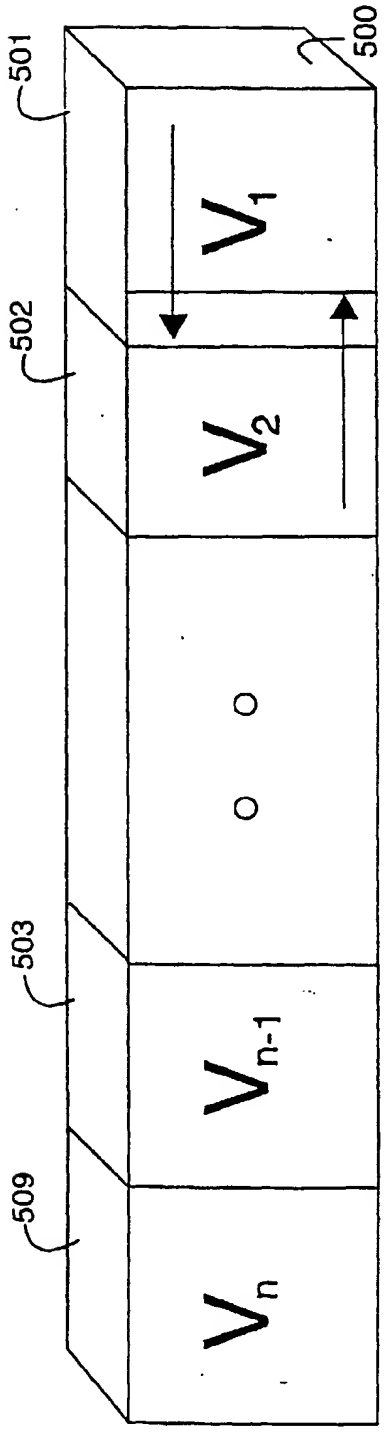

FIG. 5

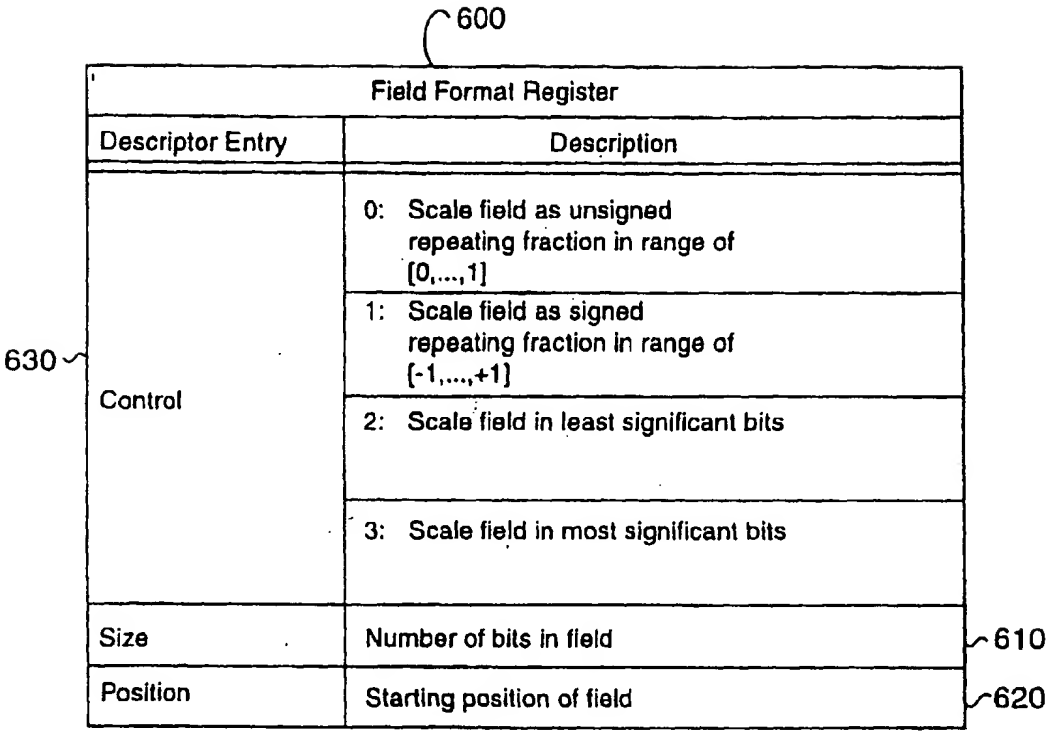
600
Field Format Register
Descriptor Entry
Description
630
Control
0: Scale field as unsigned repeating fraction in range of [0,...,1]
1: Scale field as signed repeating fraction in range of [-1,...,+1]
2: Scale field in least significant bits
3: Scale field in most significant bits
Size
Number of bits in field
610
Position
Starting position of field
620

FIG. 6

700

VoxelFormat Register.	
Field Name	Description
Field4	Descriptor of Field 4
Field3	Descriptor of Field 3
Field2	Descriptor of Field 2
Field1	Descriptor of Field 1
Field0	Descriptor of Field 0

FIG. 7

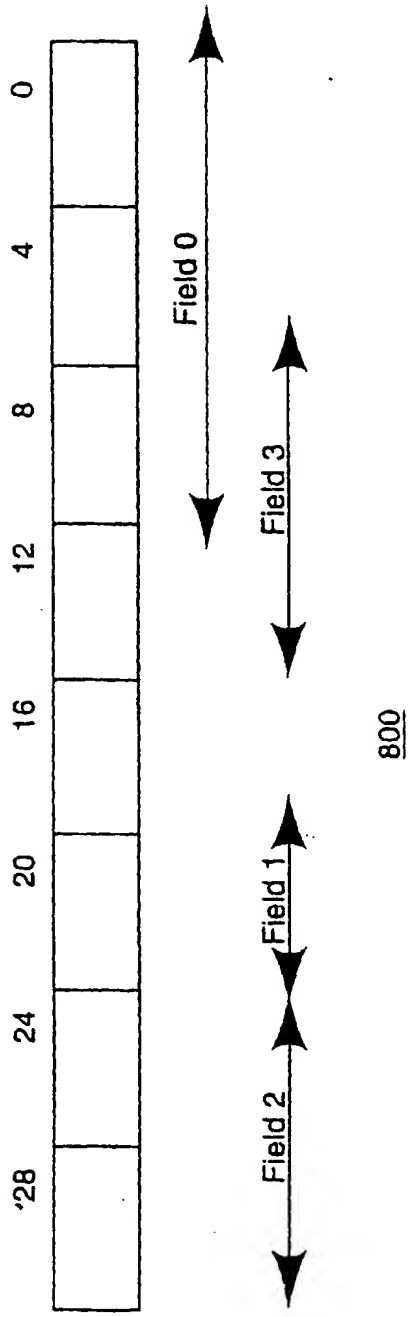

FIG. 8

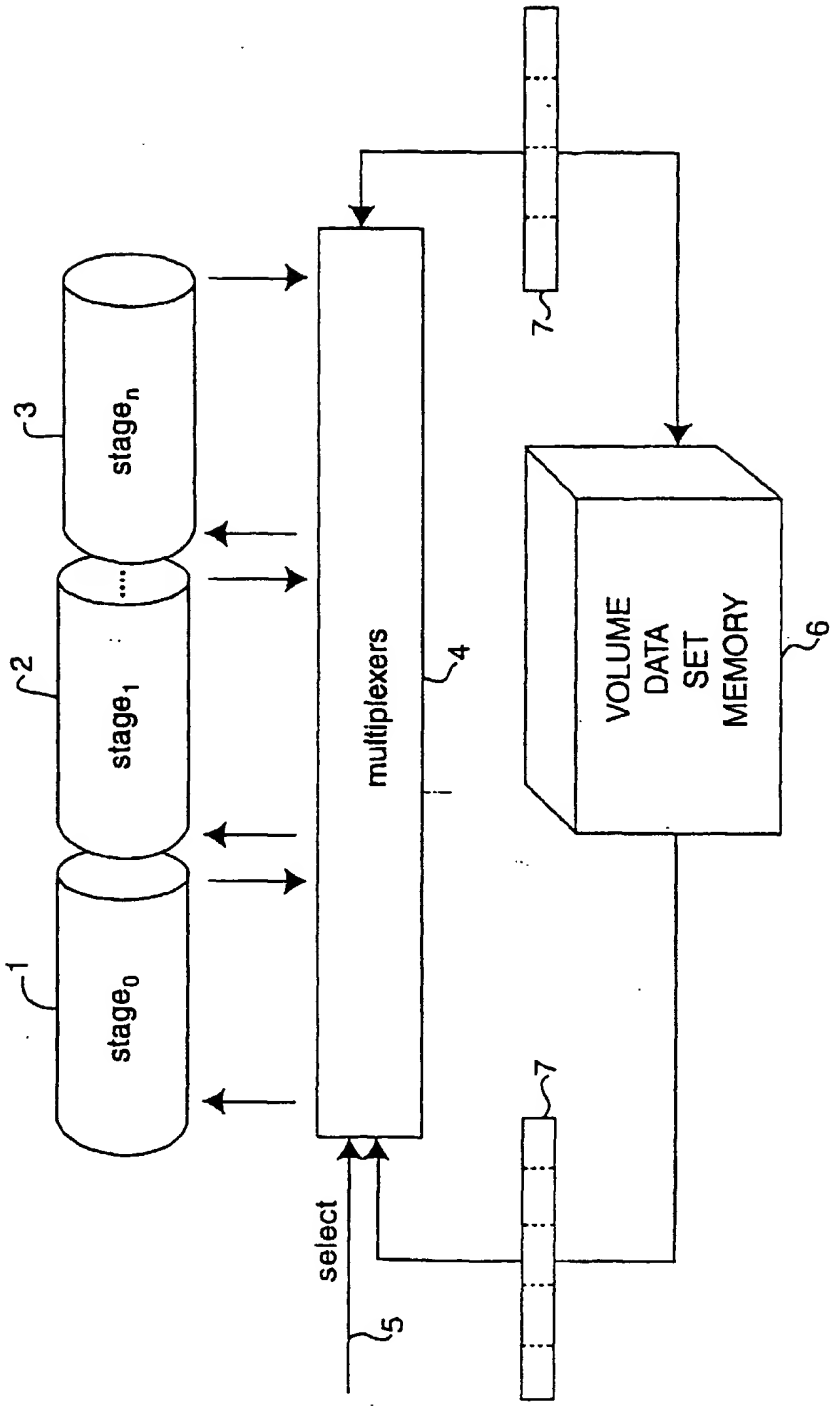
stage$_n$
stage$_1$
stage$_0$
3
2
1
multiplexers
4
VOLUME DATA SET MEMORY
6
7
7
select
5

FIG. 9

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 069 532 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
11.12.2002 Bulletin 2002/50

(51) Int Cl.[7]: **G06T 15/00**

(43) Date of publication A2:
17.01.2001 Bulletin 2001/03

(21) Application number: **00113247.1**

(22) Date of filing: **21.06.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **15.07.1999 US 353882**

(71) Applicant: **TeraRecon, Inc., A Delaware Corporation**
San Mateo, CA 94403 (US)

(72) Inventors:
- **Lauer, Hugh C.**
  Concord, MA 01742 (US)
- **Seiler, Larry D.**
  Boylston, Massachusetts 01505 (US)
- **Knittel, James M.**
  Groton, MA 01450 (US)
- **Correll, Kenneth W.**
  Lancaster, MA 01523 (US)
- **Gasparakis, Charidimos E.**
  Acton, MA 01720 (US)
- **Shimha, Vikram**
  Lexington, MA 02173 (US)
- **Bhatia, Vishal C.**
  Arlington, MA 02474 (US)

(74) Representative: **Pfenning, Meinig & Partner GbR**
Mozartstrasse 17
80336 München (DE)

(54) Multi-pass volume rendering pipeline

(57) A multi-stage pipeline renders a volume data set stored in a volume memory as samples. The pipeline includes a first stage reading the samples from the volume memory. A second stage renders the read samples, and a third stage writes the processed samples to the volume memory. The first, second and third stages use first rendering parameters during a first rendering pass and second rendering parameters during a second rendering pass to enable multi-pass volume rendering.

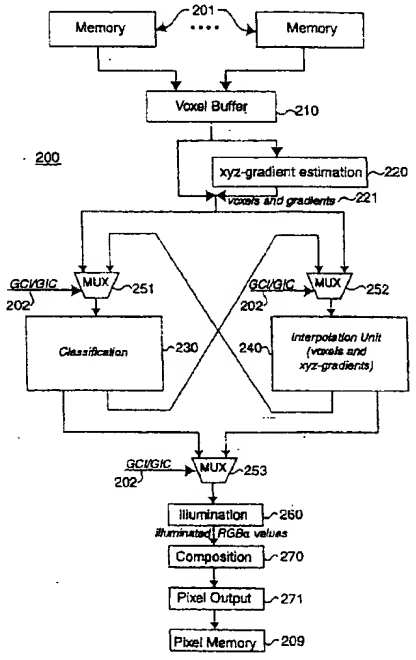

FIG. 2

EP 1 069 532 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 00 11 3247

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
A	SAKAS G ET AL: "Interactive visualization of large scalar voxel fields" PROCEEDINGS OF THE VISUALIZATION CONFERENCE. BOSTON, OCT. 19 - 23, 1992, LOS ALAMITOS, IEEE COMP. SOC. PRESS, US, vol. CONF. 3, 19 October 1992 (1992-10-19), pages 29-36, XP010029602 ISBN: 0-8186-2897-9 * page 33, right-hand column, line 41 - line 47 * * page 34, left-hand column, line 9 - line 21 * ---	1,3-5,7, 10-13,15	G06T15/00
A	STATE A ET AL: "INTERACTIVE VOLUME VISUALIZATION ON A HETERONENEOUS MESSAGE-PASSING MULTICOMPUTER" PROCEEDINGS OF THE SYMPOSIUM ON INTERACTIVE 3D GRAPHICS. MONTEREY, APR. 9 - 12, 1995, NEW YORK, ACM, US, 9 April 1995 (1995-04-09), pages 69-74, XP000546192 ISBN: 0-89791-736-7 * the whole document * ---	1,10	TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06T
A	KANUS U ET AL: "Implementations of cube-4 on the teramac custom computing machine" COMPUTERS AND GRAPHICS, PERGAMON PRESS LTD. OXFORD, GB, vol. 21, no. 2, 1 March 1997 (1997-03-01), pages 199-208, XP004064048 ISSN: 0097-8493 * the whole document * -----	1,10	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	21 October 2002	Burgaud, C

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)